Configuration Manual

MSc Research Project
Data Analytics

Rahul Jaju
Student ID: X18125671

School of Computing
National College of Ireland

Supervisor: Dr. Cristina Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rahul Jaju		
Student ID:	X18125671		
Programme:	Data Analytics	**Year:**	2018-2019
Module:	MSc Research Project		
Lecturer:	Dr. Cristina Muntean		
Submission Due Date:	12/12/2019		
Project Title:	Configuration Manual		
Word Count:	392	**Page Count:**	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:

Date: 11/12/2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	X
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	X
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	X

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

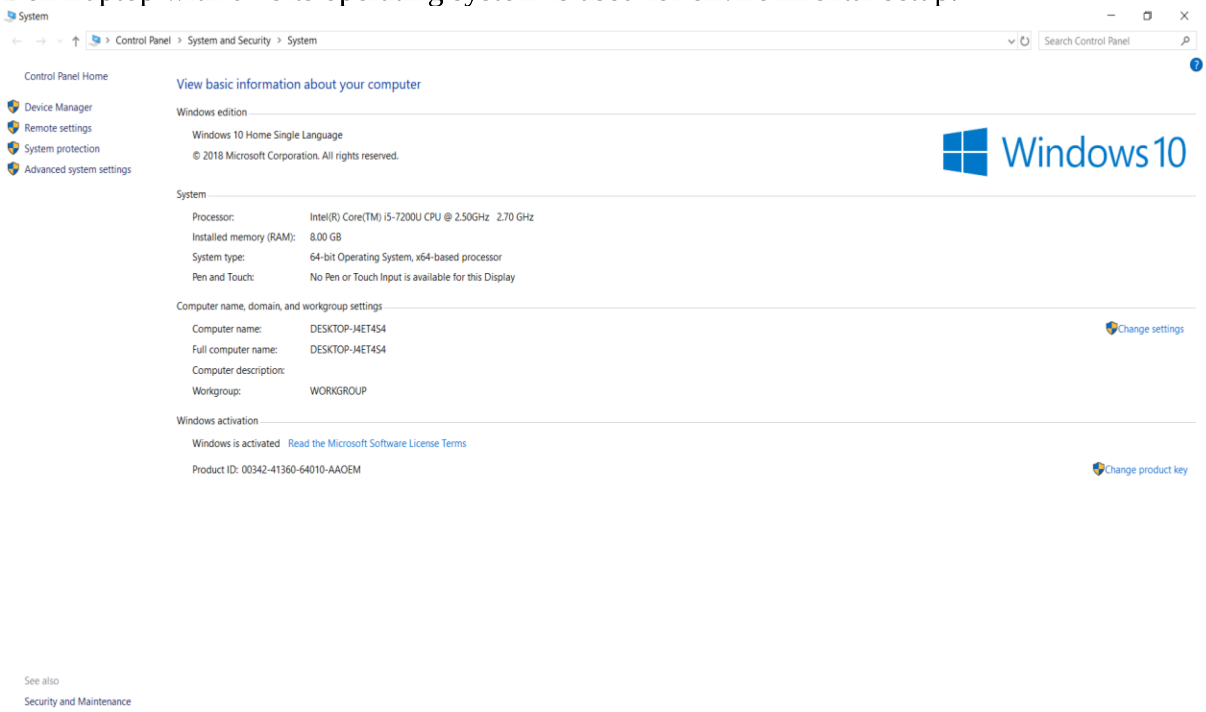
Rahul Jaju
X18125671

1 Introduction

The configuration manual quickly depicts the hardware, software and programming in Section 2 for completing the MSc Research Project titled “Classification of Traffic Signs Using Machine Learning Algorithms”. It likewise gives the subtleties on the required libraries. In section 3 there is a short depiction about information of data for all traffic sign images. The last section of this manual incorporates code and the significant output for all the execution, results and evaluation.

2 Hardware Requirement

Dell Laptop with 64 bits operating system is used for environmental setup.



2 Software Requirement

The entire implementation and the script with execution of code is done on Python version 3. Anaconda has to be installed first on system to make use of Python platform. For programming in Python Spyder IDE is utilised. We download the Anaconda from[1,2]. For installation Anaconda of version 64 bit is installed.

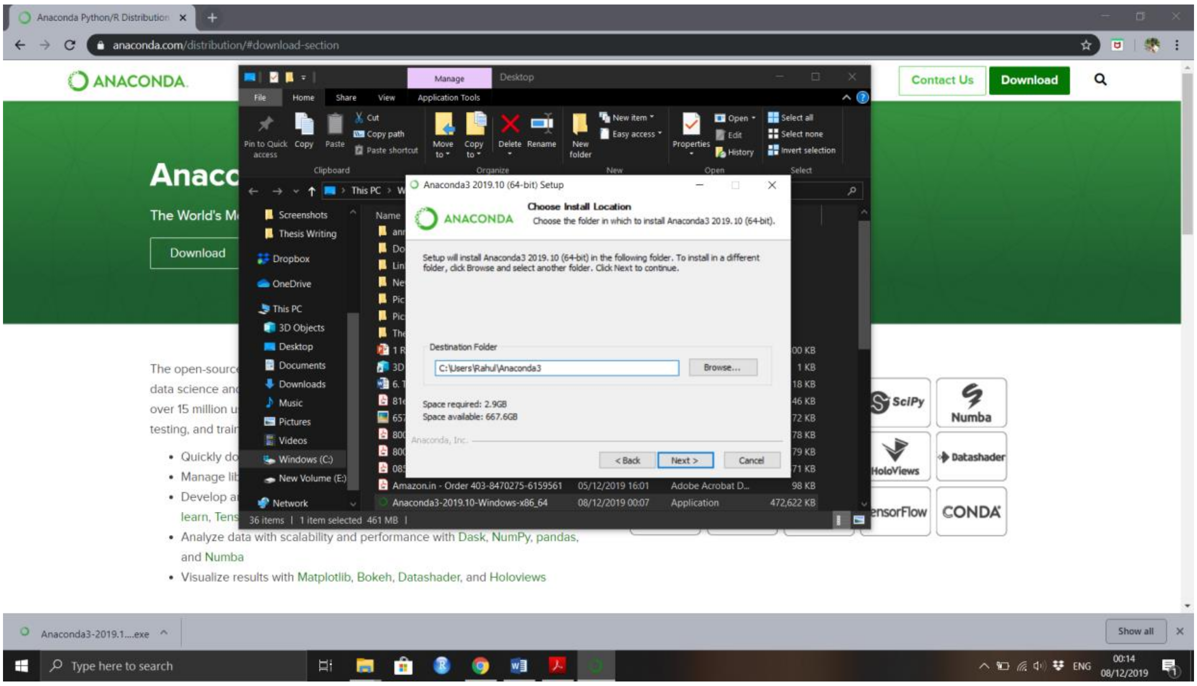


Figure 1: Downloading Anaconda

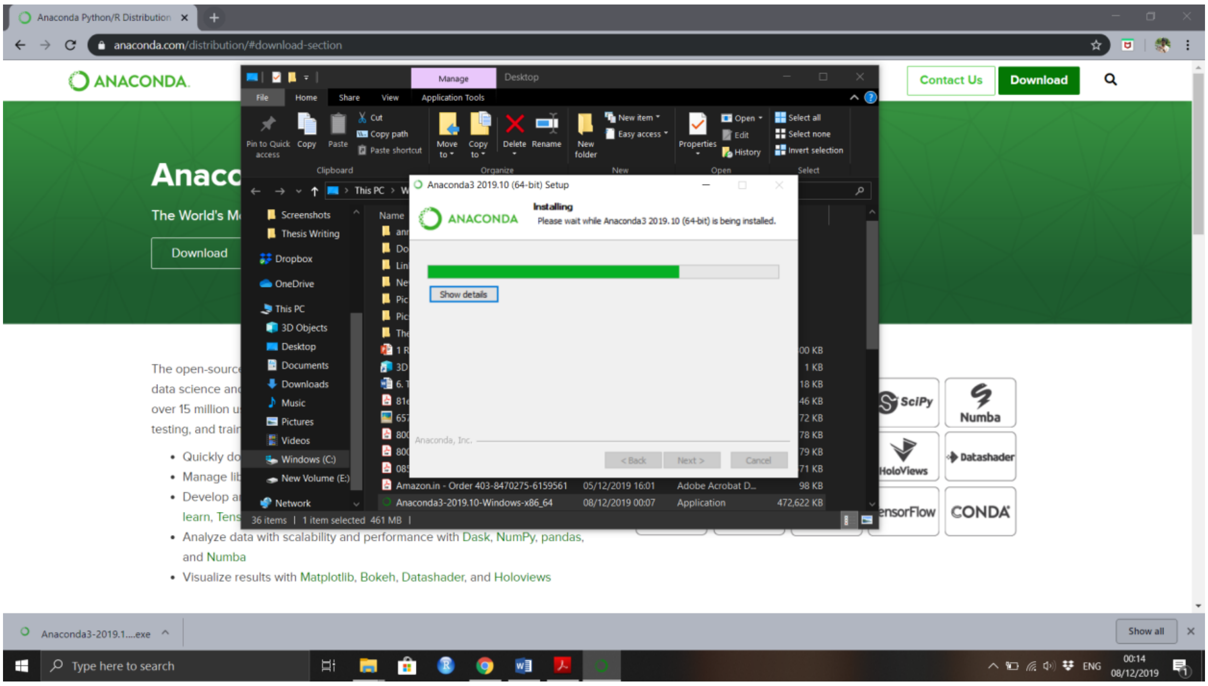


Figure 2: Installing Anaconda

We open the Anaconda Navigator from **Start** >>>> **Anaconda Navigator**

[1,2] https://www.anaconda.com/distribution/

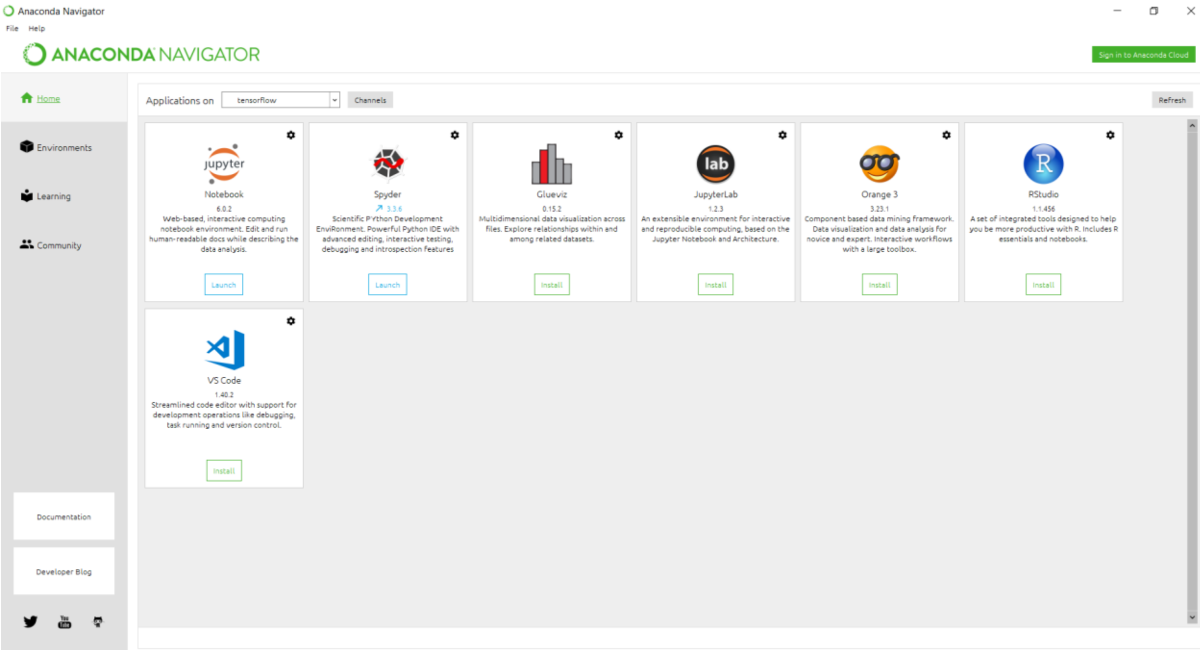


Figure 3: Anaconda Navigator

3 Data Downloading

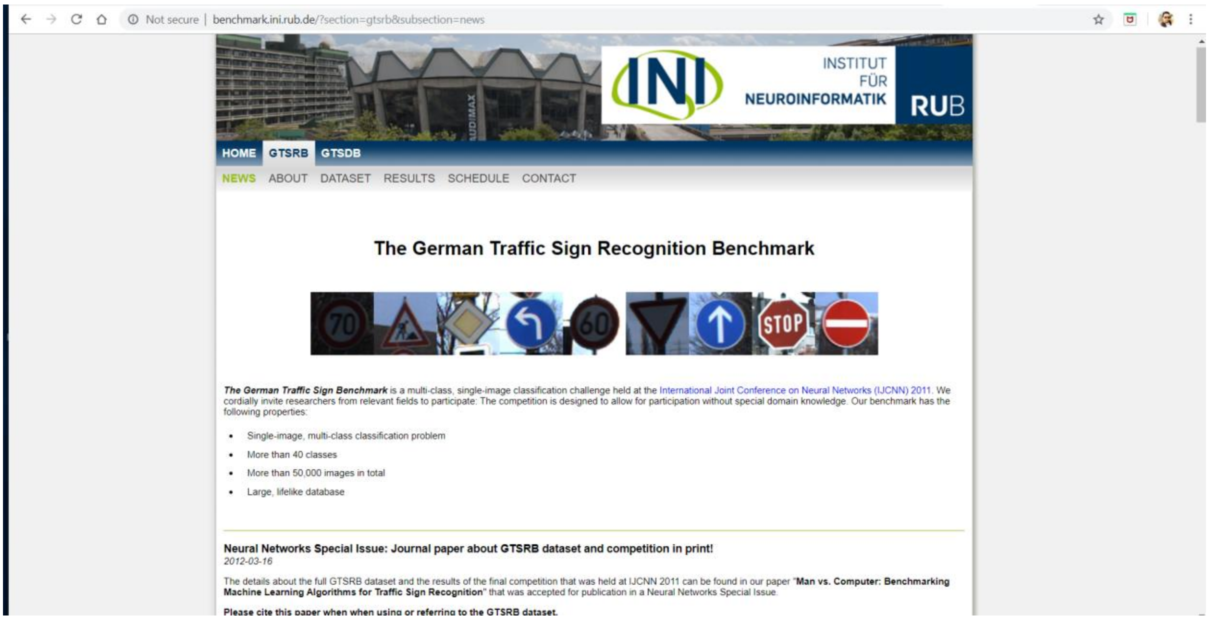


Figure 4: Downloading GTSRB data

The dataset which we have used is from German Traffic Sign Road Benchmark. The dataset consists of multi-class images of signs and can be downloaded from the link[3]

[3] http://benchmark.ini.rub.de/?section=gtsrb&subsection=news

4 Image Format Conversion

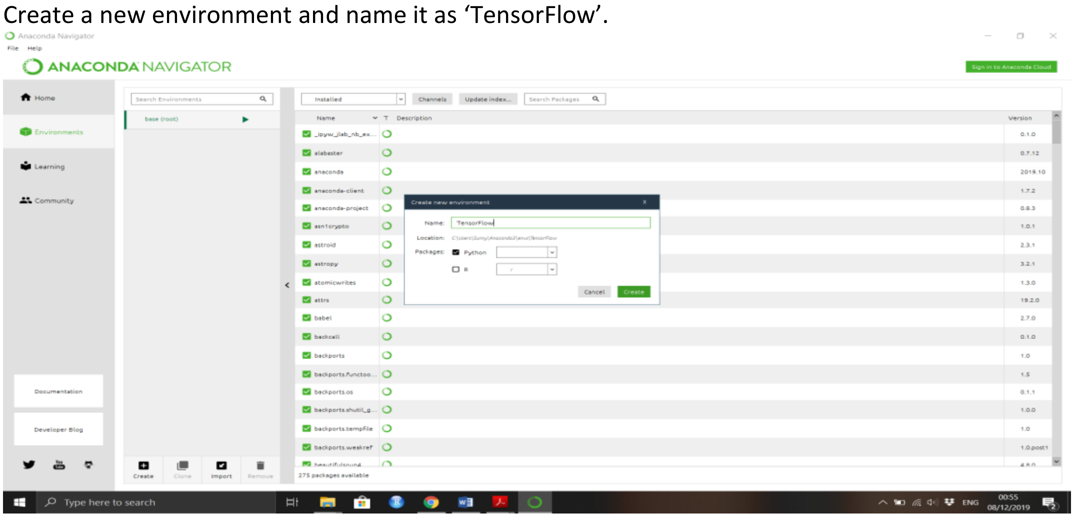
Launch Spyder and load the downloaded data. The file format of the images is in ‘**PPM**’. We convert the file format to **‘PNG’** by executing the following code.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec  3 08:50:10 2019

@author: RAHUL JAJU
STUDENT ID X18125671
"""
================================================================================
#Importing the libraries
from PIL import Image
import pandas as pd
import os

================================================================================
#Creating the directory and path of the existing dataset
def createFolder(directory):
    try:
        if not os.path.exists(directory):
            os.makedirs(directory)
    except OSError:
        print ('Error: Creating directory. ' +  directory)

image_root = 'C:/Users/Dell/Desktop/Datasets/Final_Training/Images/'
image_dst = 'C:/Users/Dell/Desktop/Datasets/Final_Training/Images/Final_Training_PNG/'
for f in range(0,43):

    # Creating another new folder where the converted files are saved
    folder = format(f, '05d')
    fol = image_dst+ format(f, '05d')
    createFolder(fol)
    # The path is given by the Meta.csv file
    csv_path = image_root + folder + '/GT-' + folder + '.csv'
    csv = pd.read_csv(csv_path, sep= ';')
    ppm_name = csv['Filename']

    # We replace the files from ppm to png and save it in the created directory
    for i in range(0, len(ppm_name)):
        im = Image.open(image_root + folder + '/' + csv['Filename'][i])
        im.save(fol+'/'+ csv['Filename'][i].replace('ppm', 'png'))
```

Figure 5: Spyder Console

5 Creating a ‘TensorFlow’ environment

Create a new environment and name it as ‘TensorFlow’.



6 Installing Essential Packages

Install the following packages in the 'TensorFlow' environment.

- ✓ Pandas
- ✓ Numpy
- ✓ Keras
- ✓ Os
- ✓ Cv2
- ✓ Skimage
- ✓ Sklearn
- ✓ Xgboost

7 Data Pre-processing

```
#Importing the required libraries
from skimage.io import imread, imshow
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import save_img
from skimage import data, exposure
from skimage.transform import rotate
from numpy import fliplr, flipud
import cv2
import matplotlib.pyplot as plt
#===========================================================================
#Setting the path for the directories=====================================
path1 = "C:/Users/Dell/Desktop/Dataset1/Train/"
path = "C:/Users/Dell/Desktop/Dataset1/Train/2/00002_00003_00029.png"
#Reading the image with its path
image = cv2.imread(path)
#Showing the image=========================================================
plt.imshow(image)
rows, cols = image.shape
#Converting from BGR to RGB color space
RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(RGB_img)
#Scaling the image by 60 percent
scale_percent = 60 # percent of original size
width = int(RGB_img.shape[1] * scale_percent / 100)
height = int(RGB_img.shape[0] * scale_percent / 100)
dim = (width, height)
img_array = img_to_array(RGB_img)
save_img('C:/Users/Dell/Desktop/Dataset1/Train/bondi_beach_grayscale.jpg', img_array)
# resize image
image_resized = cv2.resize(RGB_img, dim, interpolation = cv2.INTER_AREA)
#Graying the image
image_gray = cv2.cvtColor(image_resized, cv2.COLOR_BGR2GRAY)
#Converting from BGR to HSV
image_hsv = cv2.cvtColor(image_resized, cv2.COLOR_BGR2HSV)
#Rotating the image by 90 degree
image_rotated = rotate(RGB_img, angle=90)
#Flipping the image
image_flip = fliplr(RGB_img)
#Brightning the image
image_bright = exposure.adjust_gamma(RGB_img, gamma=0.5,gain=1)
#Darkening the image
image_dark = exposure.adjust_gamma(RGB_img, gamma=1.5,gain=1)
#Canny Edge Detector for edge detection
```

Figure 6: Data Pre-processing

```
#Graying the image
image_gray = cv2.cvtColor(image_resized, cv2.COLOR_BGR2GRAY)
#Converting from BGR to HSV
image_hsv = cv2.cvtColor(image_resized, cv2.COLOR_BGR2HSV)
#Rotating the image by 90 degree
image_rotated = rotate(RGB_img, angle=90)
#Flipping the image
image_flip = fliplr(RGB_img)
#Brightning the image
image_bright = exposure.adjust_gamma(RGB_img, gamma=0.5,gain=1)
#Darkening the image
image_dark = exposure.adjust_gamma(RGB_img, gamma=1.5,gain=1)
#Canny Edge Detector for edge detection
image_edges = cv2.Canny(image_gray,100,200)
#Plotting the images
#RGB Format
plt.subplot(121), imshow(RGB_img)
plt.title('RGB Format')
#Grayscale Format
plt.subplot(122), imshow(image_gray, cmap='gray', vmin = 0, vmax = 255)
plt.title('Grayscale Format')
#Resized Format
plt.imshow(image_resized)
plt.title('Resized format')
#HSV Format
plt.imshow(image_hsv)
plt.title('HSV Format')
#Rotated Format
plt.imshow(image_rotated)
plt.title('Rotated format')
#Flip Format
plt.imshow(image_flip)
plt.title('Flipped Format')
#Bright image Format
plt.imshow(image_bright)
plt.title('Bright Format')
#Dark Image Format
plt.imshow(image_dark)
plt.title('Dark Format')
#Canny Edge Detection Format
plt.imshow(image_edges)
plt.title('Canny Edge Detection Format')

```

Figure 7: Data Pre-processing (*continued)*

8 CNN (Convolutional Neural Network) Model

```
#Defining the CNN model=====================================================
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
model = Sequential()
# ==========================================================================

# ==========================================================================
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
#===========================================================================
#Compilation of the model
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
#===========================================================================
#using ten epochs for the training and saving the accuracy for each epoch
epochs = 50
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_val, y_val))
#===========================================================================
#Display of the accuracy and the loss values
import matplotlib.pyplot as plt
plt.figure(0)
plt.plot(history.history['acc'], label='training accuracy')
plt.plot(history.history['val_acc'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
#===========================================================================
plt.figure(1)
```


```
#Display of the accuracy and the loss values
import matplotlib.pyplot as plt
plt.figure(0)
plt.plot(history.history['acc'], label='training accuracy')
plt.plot(history.history['val_acc'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
#===========================================================================
plt.figure(1)
```


```
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
#===========================================================================
#Predicting with the test data
y_test=pd.read_csv("C:/Users/Dell/Desktop/Dataset1/Test.csv")
labels=y_test['Path'].as_matrix()
y_test=y_test['ClassId'].values

#===========================================================================
data=[]
for f in labels:
    image=cv2.imread('C:/Users/Dell/Desktop/Dataset1/Test/0/'+f.replace('Test/'
    image_from_array = Image.fromarray(image, 'RGB')
    size_image = image_from_array.resize((height, width))
    data.append(np.array(size_image))

X_test=np.array(data)
X_test = X_test.astype('float32')/255
pred = model.predict_classes(X_test)
#===========================================================================
#Accuracy with the test data
from sklearn.metrics import accuracy_score
accuracy_score(y_test, pred)
```

Figure 8: CNN Model

9 Random Forest Model

```
#==============================================================================
### Defining Random Forest Model===============================================
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()

#model fitting with the training dataset
model.fit(X_train, y_train)

#predicting with the testing set
y_pred = model.predict(X_val)

#==============================================================================
#Accuracy with the testing dataset
from sklearn import metrics
precision = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with Random Forest: {0:.2f}%".format(precision))
model.score(X_val, y_val)
# =============================================================================
# =============================================================================

# =============================================================================
# =============================================================================
#Generating Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_val, y_pred))

# =============================================================================
#
print(metrics.accuracy_score(y_pred, y_val))
#
# =============================================================================

# =============================================================================

```

Figure 9: Random Forest Model

10 Support Vector Machine Model

```
#==============================================================================
### Defining SVM Model=========================================================
from sklearn.svm import SVC
model = SVC()

#model fitting with the training dataset
model.fit(X_train, y_train)


#predicting with the testing set
y_pred = model.predict(X_val)


#==============================================================================
#Accuracy with the testing dataset
from sklearn import metrics
precision = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with SVM: {0:.2f}%".format(precision))
model.score(X_val, y_val)


#==============================================================================
#Generating Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_val, y_pred))
# print(classification_report(y_val20, y_pred20))

# =============================================================================
#
print(metrics.accuracy_score(y_pred, y_val))
# print(metrics.accuracy_score(y_pred20, y_val20))

```

Figure 10: SVM Model

11 K-Nearest Neighbour Model

```python
#==============================================================================
### Defining KNN Model=====================================================
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()

#model fitting with the training dataset
model.fit(X_train, y_train)

#predicting with the testing set
y_pred = model.predict(X_val)

#==============================================================================
#Accuracy with the testing dataset
from sklearn import metrics
precision = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with K-NN: {0:.2f}%".format(precision))

#==============================================================================
#Generating Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_val, y_pred))

# ============================================================================



```

Figure 11: KNN Model

12 Extreme Gradient Boosting Model

```
#=============================================================================
### Defining XGBoost Model====================================================
import xgboost as xgb
model = xgb.XGBClassifier(learning_rate=0.01)


#model fitting with the training dataset
model.fit(X_train, y_train)
model.score(X_val, y_val)

#predicting with the testing set
y_pred = model.predict(X_val)

#=============================================================================
#Accuracy with the testing dataset
from sklearn import metrics
precision = metrics.accuracy_score(y_pred, y_val) * 100
print("Accuracy with XGBoost: {0:.2f}%".format(precision))
model.score(X_val, y_val)

# ===========================================================================
# # =========================================================================
# ===========================================================================
#Generating Classification Report
from sklearn.metrics import classification_report
print(classification_report(y_val, y_pred))

# ===========================================================================
#
print(metrics.accuracy_score(y_pred, y_val))

```

Figure 12: XGBoost Model

References

Anaconda packages documentary
URL: *https://anaconda.org/anaconda/repo*

Guide to Anaconda environment
URL*: https://towardsdatascience.com/a-guide-to-conda-environments-bc6180fc533*

Guide to Random Forest
URL: *https://towardsdatascience.com/enchanted-random-forest-b08d418cb411*

Guide to Keras with convolutional neural network
URL: *https://towardsdatascience.com/image-recognition-with-keras-convolutional-neural-networks-/*

Guide to XGBoost with scikit-learn
URL: *https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/*

Guide to Classification Report
URL: *https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html*

Guide to Understanding for Image Classification
URL*: https://medium.com/@dataturks/understanding-svms-for-image-classification-cf4f01232700*